WE CLAIM:

1. A method of generating software test information, said method comprising the steps of:

a) generating, from a sequence of instructions, at least one of which includes a condition code, a corresponding sequence of generated instructions, for selected instructions having a condition code the corresponding generated instruction being a predetermined generated instruction having a corresponding condition code;

b) executing, on a target processor, said sequence of generated instructions; and

c) when during said step (b) said predetermined generated instruction is encountered, determining with reference to status information associated with the operation of said target processor whether the condition code of said predetermined generated instruction is satisfied and, if so, replacing said predetermined generated instruction with said corresponding instruction from said sequence of instructions so as to cause said corresponding instruction to be executed.

2. The method of claim 1, wherein each instruction of said sequence of instructions includes a condition code.

3. The method of claim 1, wherein said condition code is an instruction qualifier which prevents the instruction from being executed by said target processor unless said status information satisfies said condition code.

4. The method of claim 1, wherein said status information is predetermined architectural state associated with said target processor and said condition code specifies a status of said predetermined architectural state that must be met in order for the instruction to be executed.

5. The method of claim 1, wherein said predetermined generated instruction is an instruction which is not recognised by said target processor.

6. The method of claim 1, wherein said step a) comprises the step of:

generating, from said sequence of instructions, a sequence of generated instructions, a predetermined generated instruction being generated for each instruction in the sequence of instructions.

7. The method of claim 1, wherein said step a) comprises the steps of:

a1) partitioning said sequence of instructions into a number of instruction groups, each instruction group including one or more instructions; and

a2) generating said predetermined generated instruction for one instruction in each of said instruction group.

8. The method of claim 7, wherein said step a2) comprises the step of:

generating said predetermined generated instruction for the last instruction in each of said instruction groups.

9. The method of claim 7, wherein said predetermined generated instruction provides information relating to the number of instructions in that instruction group.

10. The method of claim 1, wherein said step c) further comprises the step of:

incrementing a coverage counter when the condition code of the predetermined generated instruction is satisfied to provide an indication that said corresponding instruction will be executed.

11. The method of claim 1, wherein said step c) further comprises the step of:

incrementing a counter associated with said corresponding instruction when the condition code of the predetermined generated instruction is satisfied to provide an indication that said corresponding instruction will be executed.

12. The method of claim 11, wherein said step c) further comprises the step of:

replacing a preceding instruction in said sequence of generated instructions with said predetermined generated instruction having a condition code corresponding to said preceding instruction.

13. The method of claim 1, wherein said step c) further comprises the step of: executing said corresponding instruction on said target processor.

14. An apparatus for generating software test information, said apparatus comprising:

instruction generation logic operable to generate, from a sequence of instructions, at least one of which includes a condition code, a corresponding sequence of generated instructions, for selected instructions having a condition code the corresponding generated instruction being a predetermined generated instruction having a corresponding condition code;

a target processor operable to execute said sequence of generated instructions and to identify the occurrence of said predetermined generated instructions; and

determination logic operable, when said predetermined generated instruction is encountered by said target processor, to determine with reference to status information associated with the operation of said target processor whether the condition code of said predetermined generated instruction is satisfied and, if so, to replace said predetermined generated instruction with said corresponding instruction from said sequence of instructions so as to cause said corresponding instruction to be executed.

15. The apparatus of claim 14, wherein each instruction of said sequence of instructions includes a condition code.

16. The apparatus of claim 14, wherein said condition code is an instruction qualifier which prevents the instruction from being executed by said target processor unless said status information satisfies said condition code.

17. The apparatus of claim 14, wherein said status information is predetermined architectural state associated with said target processor and said condition code specifies a status of said predetermined architectural state that must be met in order for the instruction to be executed.

18. The apparatus of claim 14, wherein said predetermined generated instruction is an instruction which is not recognised by said target processor.

19. The apparatus of claim 14, wherein said instruction generation logic is operable to generate, from said sequence of instructions, a sequence of generated instructions, a predetermined generated instruction being generated for each instruction in the sequence of instructions.

20. The apparatus of claim 14, wherein said instruction generation logic is operable to partition said sequence of instructions into a number of instruction groups, each instruction group including one or more instructions, and to generate said predetermined generated instruction for one instruction in each of said instruction group.

21. The apparatus of claim 20, wherein said instruction generation logic is operable to generate said predetermined generated instruction for the last instruction in each of said instruction groups.

22. The apparatus of claim 20, wherein said predetermined generated instruction provides information relating to the number of instructions in that instruction group.

23. The apparatus of claim 14, wherein said determination logic is operable to increment a coverage counter when the condition code of the predetermined generated instruction is satisfied to provide an indication that said corresponding instruction will be executed.

24. The apparatus of claim 14, wherein said determination logic is operable to increment a counter associated with said corresponding instruction when the condition code of the predetermined generated instruction is satisfied to provide an indication that said corresponding instruction will be executed.

25. The apparatus of claim 24, wherein said determination logic is operable to replace a preceding instruction in said sequence of generated instructions with said predetermined generated instruction having a condition code corresponding to said preceding instruction.

26. The apparatus of claim 14, wherein said determination logic is operable to cause the execution of said corresponding instruction on said target processor.

27. A computer program product operable, when executed on a computer, to generate software test instructions by performing the step of:

a) generating, from a sequence of instructions, at least one of which includes a condition code, a corresponding sequence of generated instructions, for selected instructions having a condition code the corresponding generated instruction being a predetermined generated instruction having a corresponding condition code.

28. The computer program product of claim 27, wherein each instruction of said sequence of instructions includes a condition code.

29. The computer program product of claim 27, wherein said condition code is an instruction qualifier which prevents the instruction from being executed by a target processor unless said status information satisfies said condition code.

30. The computer program product of claim 27, wherein said predetermined generated instruction is an instruction which is not recognised by said target processor.

31. The method of claim 27, wherein said step a) comprises the step of:

generating, from said sequence of instructions, a sequence of generated instructions, a predetermined generated instruction being generated for each instruction in the sequence of instructions.

32. The computer program product of claim 27, wherein said step a) comprises the steps of:

a1) partitioning said sequence of instructions into a number of instruction groups, each instruction group including one or more instructions; and

a2) generating said predetermined generated instruction for one instruction in each of said instruction group.

33. The computer program product of claim 32, wherein said step a2) comprises the step of:

generating said predetermined generated instruction for the last instruction in each of said instruction groups.

34. The computer program product of claim 32, wherein said predetermined generated instruction provides information relating to the number of instructions in that instruction group.

35. A computer program product operable, when executed on a computer, to generate software test information by performing the steps of:

a) executing, on a target processor, a sequence of generated instructions; and

b) when a predetermined generated instruction is encountered during said step (a), determining with reference to status information associated with the operation of said target processor whether a condition code of that predetermined generated instruction is satisfied and, if so, replacing said predetermined generated instruction with a corresponding instruction from a sequence of instructions so as to cause said corresponding instruction to be executed.

36. The computer program product of claim 35, wherein said condition code is an instruction qualifier which prevents the instruction from being executed by said target processor unless said status information satisfies said condition code.

37. The computer program product of claim 35, wherein said status information is predetermined architectural state associated with said target processor and said condition code specifies a status of said predetermined architectural state that must be met in order for the instruction to be executed.

38. The computer program product of claim 35, wherein said predetermined generated instruction is an instruction which is not recognised by said target processor.

39. The computer program product of claim 35, wherein said step b) further comprises the step of:

incrementing a coverage counter when the condition code of the predetermined generated instruction is satisfied to provide an indication that said corresponding instruction will be executed.

40. The computer program product of claim 35, wherein said step b) further comprises the step of:

incrementing a counter associated with said corresponding instruction when the condition code of the predetermined generated instruction is satisfied to provide an indication that said corresponding instruction will be executed.

41. The computer program product of claim 40, wherein said step c) further comprises the step of:

replacing a preceding instruction in said sequence of generated instructions with said predetermined generated instruction having a condition code corresponding to said preceding instruction.

42. The computer program product of claim 35, wherein said step c) further comprises the step of:

executing said corresponding instruction on said target processor.